SCHOOL OF COMPUTING (SoC)

ST0503 Back-end Web Development

2022/2023 SEMESTER 2

ASSIGNMENT 1

Instructions and Guidelines:

1. The assignment **source code** must be submitted before **Week 12, Tue, 3 Jan 2023, 08:00 a.m.** You are required to submit your source codes to the BrightSpace. Remember to provide your **Class, Group, Admission Number and Name on each file in your source code**.
2. You are required to **clearly** provide instructions on how to setup the project on the lecturer's laptop in a text file named **readme.txt** that is to be included in the softcopy submission.
3. This is an individual assessment.
4. The assignment should be implemented using Node JS, Express and MySQL.
5. The interview will be conducted during the lessons in the week 12. You are expected to explain the table relationships, program logic and modify the program during the interview. If you are absent, you will be awarded zero mark for the assignment.
6. Your application will be tested with POSTMAN.
7. **No marks will be awarded**, if the work is copied or you have allowed others to copy your work. Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence and disciplinary action will be taken against you. **If you are guilty of plagiarism, you may fail all modules in the semester, or even be liable for expulsion**.

8. 50% of the marks will be deducted for assignments that are received within ONE (1) calendar day after the submission deadline. No marks will be given thereafter. Exceptions to this policy will be given to students with valid LOA on medical or compassionate grounds. Students will need to inform the lecturer as soon as reasonably possible. Students are not to assume on their own that their deadline has been extended.

Assignment 1: DVD rental Store

Background

SP DVD would like to open a few DVD rental store. As such, SP DVD has tasked you to design the backend API specs for SP DVD. There exist a database which the store currently use with the Point-of-sales system. You are to make use of the existing database and development a backend API to support functionalities such as retrieve of customers' records, creation of new data records.

Assignment Basic Requirements

You are required to fulfil the following basic requirements:

- Create a new MySQL database named **bed_dvd_db** and import the data with the provided sql file **sakila_bed.sql**, you are not to modify the table structure nor their relational settings.
- Username for accessing the database must be **bed_dvd_root**
- Password for accessing the database must be **pa$$woRD123**
- Study and understand the relationships between tables.
- Create an Express server that comply with the specs provided.
- Consume data from MySQL using the mysql library.

Grading Guidelines

The assignment will be assessed based on the following criteria:

- Demonstrate and satisfy the web api endpoint functionalities listed below to access/update and return relevant data from the database upon success or failure
  Note: There are minimum of **8** APIs end-points to complete. Components of grading per api is based primarily on correctness (returning the right json data with proper Model, Controller layer and database calls) and returning the right failure message.
- Proper understanding of the imported database design, students are expected to explain the table relationship during interview.
- Code should have proper indentation and comments
- Question & Answer during the interview
- A word document containing screenshots of the testing you have done with postman for every endpoint you coded.

Note: The assessments of the assignment will also take into consideration your dependency on your module tutor while working on the assignment.

DVD Rental Store

For this assignment, you are required to create the following endpoints. The response message body for failed operations can be determined by you in the API.

Endpoint 1:

GET	/actors/{actor_id}
	Return actor information of the given actor_id.

Parameters		
Name	Type	Description
actor_id	path	Record id of an actor

Responses	Content Type: application.json
Code	Description
200	Example call: /actors/56 Example Value: `{ "actor_id": "56", "first_name": "DAN", "last_name": "HARRIS", }`
204	Example call: /actors/9999 No Content. Record of given actor_id cannot be found.
500	Example Value: `{ "error_msg": "Internal server error" }`

Endpoint 2:

GET	/actors
	Return the list of actors ordered by first_name, limited to 20 records offset 0 by default.

Parameters		
Name	Type	Description
limit	query	Default to 20 if not provided. Determines the number of records to be returned
offset	query	Default to 0 if not provided. Determines the number of records to skip before starting to collect the result set

Responses	Content Type: application.json
Code	Description

200	Example call: /actors?limit=3&offset=2
	
	Example Value:
	`[ { "actor_id": "165", "first_name": "AL", "last_name": "GARLAND" }, { "actor_id": "173", "first_name": "ALAN", "last_name": "DREYFUSS" }, { "actor_id": "146", "first_name": "ALBERT", "last_name": "JOHANSSON" } ]`
	
	No record found:
	Example call: /actors?limit=10&offset=5000
	
	Example Value:
	`[]`

500	Example Value: `{ "error_msg": "Internal server error" }`

Endpoint 3:

POST	/actors
	Add a new actor to the database (note: actors can have the same first_name and last_name)

Parameters

Name	Type	Description
object	body	Example Value: `{ "first_name": "KEN", "last_name": "LEE" }`

Responses	Content Type: application.json
Code	Description
201	Record was created successfully Example Value: `{ "actor_id": "202" }` Note: the actor_id is an auto increment number.
400	Missing information from body object. Example Value: `{ "error_msg": "missing data" }`
500	Example Value: `{ "error_msg": "Internal server error" }`

Endpoint 4:

PUT	/actors/{actor_id}
	Update actor's first name or last name or both.

Parameters

Name	Type	Description
actor_id	path	Record id of an actor
object	body	Example Value: `{ "first_name": "KENNY" }`

Responses	Content Type: application.json
Code	Description
200	Record was updated successfully Example Value: `{ "success_msg": "record updated" }`
204	No Content. Record of given actor_id cannot be found.
400	Missing information from body object. Example Value: `{ "error_msg": "missing data" }`
500	Example Value: `{ "error_msg": "Internal server error" }`

Endpoint 5:

DELETE	/actors/{actor_id}
	Remove actor from database.

Parameters		
Name	Type	Description
actor_id	path	Record id of an actor

Responses	Content Type: application.json
Code	Description
200	Record was deleted successfully Example Value: `{ "success_msg": "actor deleted" }`
204	Record of given actor_id cannot be found. No Content. Record of given actor_id cannot be found.
500	Example Value: `{ "error_msg": "Internal server error" }`

Endpoint 6:

GET	/film_categories/{category_id}/films
	Return the film_id, title, rating, release_year and length of all films belonging to a category.

Parameters

Name	Type	Description
category_id	path	Category id of films

Responses — Content Type: application.json

Code	Description

200	Example call: /film_categories/8/films
	
	Example Value:
	[
	{
	"film_id": "5",
	"title": "AFRICAN EGG",
	"category": "Family",
	"rating": "G",
	"release_year: "2006",
	"duration": "130"
	},
	{
	"film_id": "31",
	"title": "APACHE DIVINE",
	"category": "Family",
	"rating": "NC-17",
	"release_year: "2006",
	"duration": "92"
	},
	{
	"film_id": "43",
	"title": "ATLANTIS CAUSE",
	"category": "Family",
	"rating": "G",
	"release_year: "2006",
	"duration": "170"
	}
	]
	Note: assuming there are 3 films in the category with category_id 8
	
	No record found: Example call: /film_categories/9999/films
	Example Value:
	[]

500	Example Value:
	{
	"error_msg": "Internal server error"
	}

Endpoint 7:

GET	/customer/{customer_id}/payment
	Return the payment detail of a customer between the provided period.

Parameters		
Name	Type	Description
customer_id	path	customer id
start_date	query	Start date of the query period in the format "yyyy-mm-dd"
End_date	query	End date of the query period in the format "yyyy-mm-dd"

Responses	Content Type: application.json
Code	Description

200	Example call: /customer/5/payment?start_date=2005-06-01&end_date=2005-06-30
	
	Example Value:
	{
	"rental": [
	{
	"title": "REEF SALUTE",
	"amount": "3.99",
	"payment_date": "2005-06-15 22:03:14"
	},
	{
	"title": "LUCK OPUS",
	"amount": "2.99",
	"payment_date": "2005-06-16 08:01:02"
	},
	{
	"title": "VIDEOTAPE ARSENIC",
	"amount": "4.99",
	"payment_date": "2005-06-17 15:56:53"
	},
	{
	"title": "DOUBLE WRATH",
	"amount": "2.99",
	"payment_date": "2005-06-19 04:20:13"
	},
	{
	"title": "DURHAM PANKY",
	"amount": "4.99",
	"payment_date": "2005-06-20 18:38:22"
	},
	],
	"total": "19.95"
	}

	No payment details found for the given period:
	
	Example Value:
	<pre>{
	  "rental": [],
	  "total": "0"
	}</pre>

500	Example Value:
	<pre>{
	  "error_msg": "Internal server error"
	}</pre>

Endpoint 8:

POST	/customers
	Add a new customer to the database (note: customer's email address is unique)

Parameters

Name	Type	Description
object	body	Example Value: `{ "store_id": "1" "first_name":"MARTIN", "last_name": "SIM", "email":"martin_sim@gmail.com"; "address": { "address_line1": "77 elm street", "address_line2": "", "district": "California", "city_id" "449", "postal_code": "17886", "phone": "6325-8596", } }`

Responses	Content Type: application.json
Code	Description
201	Record was created successfully Example Value: `{ "customer_id": "856" }` Note: the customer_id is an auto increment number.
400	Missing information from body object. Example Value: `{ "error_msg": "missing data" }`
409	Conflict, user try to create a record with duplicate email address. Example Value: `{ "error_msg": "email already exist" }`
500	Example Value: `{ "error_msg": "Internal server error" }`

Additional End-points

A maximum of 2 additional End-points:

- Only POST or PUT end-points allowed
- Each end-points must involve at least 2 related tables
- Document your end-point (refer to end-point 1 to 7), in a Word document file named ***Additional_EndPoint.docx*** that is to be included in the softcopy submission.

- END -